

Isabel Leonard

Medical and Technical Translation

5 Hearn Street, Watertown, MA 02472-1502, USA

Phone 617-661-3273 Fax 253-595-9305

e-mail: isabelleleonard@comcast.net

10/577005

APR 15 2006 17:00 PTO 24 APR 2006

File 3000-52

VERIFICATION OF TRANSLATION

I hereby declare and state that I am knowledgeable of each of the German and English languages and that I made and reviewed the attached translation of a patent application entitled "Method of Storing Data in a Random Access Memory, and an Encryption and Decryption Device" from the German language into the English language, and that I believe my attached translation to be accurate, true, and correct to the best of my knowledge and ability.

Date: April 14, 2006



Isabel A. Leonard

Translator

Description

Method of Storing Data in a Random Access Memory, and an Encryption and Decryption Device

This invention relates to a method of storing data in a random access memory, and to an encryption and decryption device.

In order to ensure data security or to protect copyrights, a known approach is to store the data in encrypted form in a read-only memory (ROM), such as, for example, an EPROM, EEPROM, CD-ROM, DVD-ROM, etc. These data may relate to both data from executable programs (program codes) as well as video or audio data. An approach is also known whereby video data or audio data are transmitted in encrypted form from a transmitting device to a receiving device.

The objective is that the use of the encryption-stored or encryption-transmitted data is thereby enabled only for those users who have a corresponding decryption unit (decoder) with a "matching" key.

Conventional encryption algorithms, such as, for example, the DES method (DES = Data Encryption Standard) or the AES method (AES = Advanced Encryption Standard) encrypt/encode the data blockwise, wherein with the DES method, for example 64 data bits are encoded in one block. Since in this method the number of data bits contained in a data block is usually greater than the number of data bits of a data word processable by a processing unit, it is necessary to have the processing unit first store the data words obtained after decoding a data block in a random access memory (RAM) before these undergo further processing.

These RAMs located externally to the processing unit represent a security risk insofar as there is a possibility that the encrypted data can be tapped along the link between the RAM and the processing unit. These data, for example video or audio data, can then be stored in unencrypted form, thereby making them accessible to unauthorized use.

If the data stored in the RAM happen to be the data of a program code, then there is the risk that the program flow may be determined by unauthorized persons. In addition, there is the risk that unauthorized program code may be fed into the unit executing the program in order, for

example, to provide additional functions which are not supposed to be provided by the authorized program code.

The goal of this invention is to provide a secure method of storing data in a RAM which does not have the afore-mentioned disadvantages and is implementable at low cost, as well as a device to encrypt/decrypt the data stored in a RAM.

These goals are achieved by a method according to Claim 1 and by a device according to Claim 12. Advantageous embodiments of the invention are described in the subordinate claims.

In the method according to the invention for storing data in a random access memory (RAM) in which data words are storable with a predetermined number of data bits, an encryption of each data word is effected before storage whereby a permuted data word with a predetermined number of data bits is generated from each data word, or from a data word derived from this data word, by one-to-one rearrangement/permutation of the individual data bits using a first permutation key.

An advantageous aspect of this method is that the individual data bits of the permuted data word are substituted using a first substitution key before the storage, wherein the data word encrypted by permutation and subsequent substitution is stored in the memory. In this connection, there is also the possibility of substituting the data bits of the data word to be encrypted before the permutation using a first substitution key, and of storing the data word obtained from the substitution and subsequent permutation as the encrypted data word.

The encryption of the individual data words is preferably effected in the same chip in which the processing unit processing the data words is integrated. The data words transferred externally from this chip to the RAM memory for storage are provided in encrypted form in this method, and are thus protected against interference effects or unauthorized tapping of the data. In this method, the encryption is effected data word by data word, with the result that, unlike the case of blockwise encryption, no additional storage on the chip is required for the encryption or a decryption.

The permutation or rearrangement of the individual data bits as determined by the permutation key represents an effective encryption method. Given a data word of 32 bit width, there are $32! \approx 2,6 \cdot 10^{35}$ different permutation possibilities. This number of permutation possibilities for a data word of 32 bit length increases by a factor of 2^{32} when in addition to the permutation a sub-

stitution of the input data word, or of the already permuted data word, is effected using a substitution key of 32 bit length.

The substitution of a data word to be substituted is effected as determined by the substitution key, for example by assigning a key bit of the substitution key to each data bit of the data word, wherein the respective data bit is mapped, in unchanged or inverted form as a function of the value of the assigned substitution key bit, to the data word resulting from the substitution.

In one embodiment, the permutation key comprises a number of unique subkeys corresponding to the number of the data bits of the data word to be permuted, these keys each being assigned to a data bit of the data word resulting from the permutation. The individual subkeys indicate which of the data bits of the data word to be permuted is to be mapped to the respective data bit to which the subkey is assigned.

Each subkey of the permutation key here comprises a number of key bits, wherein preferably provision is made to implement incrementally the mapping of a data bit of the data word to be permuted to a data bit of the permuted data word using a subkey according to the following steps:

- a) selecting a first group of data bits from the data bits of the permuted data word as determined by a first key bit of the subkey;
- b) selecting a second group of data bits from the first group of data bits obtained by the previous selection as determined by a second key bit of the subkey;
- c) repeating step b), each time using an additional key bit in order to select from the group obtained by the previous selection an additional group until the selected group comprises only one more data bit which corresponds to the data bit of the permuted data word.

This type of incremental selection procedure to map a data bit of the data word to be permuted to a data bit of the permuted data word provides the advantage that no storage elements are required to implement it.

The permutation key, and possibly the substitution key, are regenerated before a new writing to the RAM memory, for example, after connection to a device containing the RAM memory.

The substitution key, which comprises a number of substitution key bits, corresponding to the number of data bits is generated here by picking out a corresponding number of bits from a sequence supplied by a random number generator.

When generating the permutation key, care must be taken that the individual subkeys differ so as to ensure a one-to-one assignment of a data bit of the data word to be permuted to a data bit of the permuted data word. In order to generate the individual sub-permutation-keys which are each assigned to a bit position of the permuted data word, and which together yield the permutation key, provision is made to generate a sub-permutation-key consecutively for each bit position of the permuted data word, and thereby to check whether the generated sub-permutation-key has already been generated for another bit position. If this sub-permutation-key has already been generated, it is rejected and a new sub-permutation-key is randomly generated for the given bit position. If the randomly generated sub-permutation-key does not yet exist, then this key is retained for the given bit position. This procedure repeats until to each bit position of the permuted data word one sub-permutation-key has been assigned for the selection of a data bit of the data word to be permuted.

The decryption of the data words stored in the RAM is effected analogously to the encryption procedure. If in a two-step procedure comprising permutation and substitution the data word to be encrypted is first permuted and then substituted, then during decryption the encrypted data word is first “back”-substituted using a second substitution key to undo the substitution effected during encryption, and subsequently “back”-permuted using a second permutation key in order to undo the permutation effected during the encryption.

If during encryption of the data word first a substitution and then a permutation is effected, then during decryption the encrypted data word is first permuted using the second permutation key, then substituted in order to recover the original data word.

Depending on the type of substitution used, the first substitution key can be selected in identical form to the second substitution key, for example, whenever the substitution consists in mapping the individual data bits unchanged or inverted as determined by the key bits of the substitution key.

The following employs embodiments to explain the invention in more detail based on the figures.

Figure 1 shows an arrangement comprising an encryption and decryption arrangement which encrypts the data to be stored in a random access memory and which decrypts the data read out from the random access memory.

Figure 2 shows an embodiment of an encryption and decryption arrangement comprising an encryption unit, a decryption unit, a key generator, and a random number generator.

Figure 3 shows an embodiment of an encryption arrangement which comprises a permutation unit and a substitution unit.

Figure 4 schematically illustrates the structure of a permutation unit which comprises selection units.

Figure 5 shows an embodiment of a selection unit which comprises multiple selection stages with selection switches.

Figure 6 illustrates the functional principle of a selection unit for a data word of 8 bit width.

Figure 7 shows the circuit-logic-implemented embodiment of the selection switches shown in Figure 5.

Figure 8 schematically illustrates an embodiment of the substitution unit shown in Figure 3, the substitution unit comprising multiple substitution elements.

Figure 9 illustrates a possible embodiment of the substitution elements shown in Figure 8.

Figure 10 illustrates the construction of the permutation key from subkeys and key bits, and the construction of the substitution key.

Figure 11 illustrates the complete structure of a permutation unit for an encryption unit as indicated in Figure 2 for data words of 4 bits.

Figure 12 shows the permutation unit corresponding to the permutation unit shown in Figure 11 for use in a decryption unit as indicated in Figure 2.

Figure 13 schematically illustrates the structure of an internal memory, provided in the key generator, to store a first permutation key for the encryption and a second permutation key for the decryption.

Unless otherwise indicated, identical reference notations in the figures denote components and signals of identical meaning.

Figure 1 shows a random access memory (RAM) 20 which is designed to store data words of n-bit length. Memory 20 has an input 21 to read in data words to be stored, and an output 22 to read out stored data words. Not shown in Figure 1 are the required control wires through which the memory addresses are communicated to the memory, at which addresses the

individual data words are to be stored or from which addresses the individual data words are to be read out.

Processing of the data words read into memory 20, or read out of this memory, is effected in a data processing unit 30, for example, a processor. Depending on the type of this processor, the data words stored in memory 20 are, for example, data words of a program code which is executed by the processor, or data words of video or audio data which are moved by processor 30 through suitable output units in order to be perceived.

Data processing unit 30 and memory 20 are not integrated on a common chip, as indicated in Figure 1 by the broken line between data processing unit 30 and memory 20. In order to prevent any "wiretapping" of or interference with data communication between data processing unit 30 and memory 20, an encryption and decryption unit 10 is provided between data processing unit 30 and memory 20 on the same chip on which data processing unit 30 is located. This device 10 encrypts data words M outputted by data processing unit 30 so as to provide encrypted data words M' which are stored word-by-word in memory 20. In the reverse direction, device 10 decrypts data words M' stored in encrypted form in memory 20 in order to recreate the original data word processable by data processing unit 30. In Figure 1 and subsequently, M denotes an arbitrary unencrypted data word of length n , while M' denotes an arbitrary encrypted data word of length n generated by encrypting a data word M .

Figure 2 schematically illustrates the structure of this encryption and decryption device 10. The device shown comprises an encryption unit 11 which has an input of n -bit width to supply an unencrypted data word M , and an output 111 to output an encrypted data word M' . Encryption of data word M is effected as determined by a first key C which is provided by a key generator 13. For the purpose of supplying this first key C , a binary random sequence RS is fed by a binary random number generator 12 to key generator 13.

Device 10 further comprises an encryption unit 11' with an input 110' to supply an encrypted data word M' of n -bit width, and an output 111' to supply the decrypted data word M generated from encrypted data word M' . The decryption is effected as determined by a second key C' which is matched to first key C and which is also provided by key generator 13.

The decryption unit maps the data word using first key C uniquely to the encrypted data word M' , wherein:

$$M' = E(M, C) \quad (1),$$

where E stands for the encryption function implemented by encryption unit 11. Analogously:

$$M = D(M', C') \quad (2),$$

where D stands for the decryption function implemented by decryption unit 11'.

Figure 3 schematically illustrates an embodiment of encryption unit 11 which in the example comprises a permutation unit 14 and a substitution unit 15. Permutation unit 14 has inputs to supply the individual data bits $M[n-1] \dots M[0]$ of data word M, and outputs to supply data bits $Mp[n-1], Mp[k], Mp[0]$ of a permuted data word Mp. The individual data bits $Mp[n-1] \dots Mp[0]$ of permuted data word Mp result from the data bits $M[n-1] \dots M[0]$ of data word M by permutation/rearrangement as determined by a permutation key P. The permutation here is effected on a one-to-one basis, that is, one data bit each of unencrypted data word M is mapped to one data bit of permuted data word Mp.

In the example, data bits $Mp[n-1] \dots Mp[0]$ of permuted data word Mp are then substituted by a substitution unit 15 as determined by a substitution key S, wherein substitution unit 15 provides the data bits of encrypted data word M'. As determined by substitution key S, one data bit each of permuted data word Mp is mapped by substitution unit 15 to one data bit $M'[n-1] \dots M'[0]$ of encrypted data word M'.

The following explains the structure and the functional principle of permutation unit 14 based on Figures 5 through 7. Next, the structure and functional principle of substitution unit 15 will be explained based on Figures 8 and 9.

With reference to Figure 4, permutation unit 14 has a number of selection units $14_n-1 \dots 14_0$ corresponding to the number of data bits of the data word to be encrypted M, wherein all the data bits $M[n-1] \dots M[0]$ of data word to be encrypted M are supplied to each of these selection units, and wherein the individual selection units $14_n-1 \dots 14_0$ each provide a data bit $Mp[n-1] \dots Mp[0]$ of the permuted data word Mp. Mapping of one of the data bits of unencrypted data word M to one of the data bits of permuted data word Mp is effected in selection units $14_n-1 \dots 14_0$ as determined by sub-permutation-keys $P[n-1], P[k], P[0]$. Each of these sub-permutation-keys differ in order to map each of the data bits of input data word M exactly once to a data bit of permuted data word Mp. The sub-permutation-keys together produce the permutation key, where:

$P = (P[n-1], \dots, P[0])$.

The individual selection units $14_{n-1} \dots 14_0$ are structured identically, the structure of a random one of these selection units, here selection unit 14_k , being explained below based on Figure 5.

This selection unit 14_k provides the data bit $M_p[k]$ from the data bits $M[n-1] \dots M[0]$ of data word M as determined by sub-permutation-key $P[k]$. This sub-permutation-key comprises m key bits $P[k, m-1] \dots P[k, 0]$.

Selection unit 14_k comprises multiple selection stages $141_0 \dots 141_{m-1}$. All the data bits of input data word M are supplied to a first selection stage 141_0 . As determined by a first key bit $P[k, 0]$ of sub-permutation-key $P[k]$, this first selection stage 141_0 selects a first group of data bits which are supplied to a second selection stage 141_1 . As determined by a second key bit $P[k, 1]$, second selection stage 141_1 generates from this first group a second group which is supplied to the third selection unit 141_2 .

In the example shown, reduction of the data bits present in the respective groups is effected from selection stage to selection stage by a factor of 2, such that after $m = \log_2(n)$ selection stages only one data bit is left which corresponds to data bit $M_p[k]$ of permuted data word M_p . In this example in which $n = 32 = 2^5$, there are thus $m = 5$ selection stages.

In the example, each of the selection stages comprises a number of selection switches 142, to which two data bits each of a data group are supplied, and which, as determined by a permutation key bit, select one of the two data bits and pass it on to the next selection stage.

The supply of the individual data bits to the selection switches of the given selection stage is effected such that two data bits each are supplied to a selection switch, which data bits have successive bit positions in relation to the group from which the given selection stage has made a selection. In the example of Figure 5, the respective higher-order bit is supplied to a first input IN1, while the respective lower-order bit is supplied to a second input IN2, of the given selection switch 142. In the example shown, given a key bit "1", the bit applied at input IN1, that is, the higher-order bit, is passed to output OUT1, and thus to the next selection stage.

The functional principle of the selection stage shown in Figure 5 is explained below based on an 8-bit-wide data word M in Figure 6. From these 8 data bits $M[7] \dots M[0]$, one is selected to generate data bit $M_p[k]$ of the permuted data word. The first key bit $P[k, 0]$ of subkey $P[k]$ has a value of 1 so that out of two data bits that are consecutive in terms of significance the

higher-order one is selected, thus yielding a first group with data bits $M[7]$, $M[5]$, $M[3]$, $M[1]$. Out of each two consecutive, in terms of their significance, data bits, that is, data bits $M[7]$, $M[5]$ and $M[3]$, $M[1]$, one data bit each is selected as determined by the second key bit $P[k,1]$. In the example, this key bit is “0”, so that in each case the lower-order one of the two data bits is selected, that is, data bits $M[5]$, $M[1]$. Out of this resulting additional group of data bits, one, in this case the higher-order one or data bit $M[5]$, is selected as determined by the third key bit $P[k,2]$ in order to generate data bit $M_p[k]$ of the permuted data word.

If one arranges the data bits in each of the selection groups as a function of their significance, and out of two adjacent ones in terms of their significance given a key bit “1” one selects the higher-order data bit, and given a key bit “0” one selects the lower-order one of these two data bits, then the value of the bit position of the selected data bit, in this case of data bit $M[5]$, corresponds to the decimal equivalent of subkey $P[k]$, as explained below:

If one views subkey $P[k]$ as a binary numerical sequence, the most significant bit (MSB) of which is generated by the key bit $P[k,m-1]$ of the last selection stage, and the least significant bit (LSB) of which is generated by key bit $P[k,0]$ of the first selection stage, then the decimal equivalent of this binary sequence, in this case $101_2 = 5_{10}$, corresponds to the bit position of data bit $M[5]$ selected from data word M .

A circuit-logic implementation of one embodiment of one of the selection switches 142 is shown in Figure 7. In order to implement the described selection function, the selection switch comprises two AND gates, AND1, AND2, the outputs of which are supplied to an OR gate OR1, wherein the output of this OR gate forms the output OUT1 of the selection switch. One each of inputs IN1, IN2 to supply the data bits is supplied to one of the AND gates AND1, AND2. The other input of the AND gate AND1 is coupled to the third input IN3 to supply a key bit, wherein this key bit is supplied in inverted form through an inverter INV1 to the other input of AND gate AND2. When a logical “1” is applied at the third input IN3, the data bit applied at first input IN1 is passed through the first AND gate AND1 and OR gate OR1 to output OUT1. Given a logical “0” at the third input IN3, the data bit at second input IN2 is accordingly passed through second AND gate AND2 and OR gate OR1 to output OUT1.

With reference to Figure 8, substitution unit 15 comprises a number of substitution elements $15_{n-1} \dots 15_0$ corresponding to the number of data bits, one data bit of the data word to be substituted being supplied to each of the elements; in the example of Figure 3, that of permuted

data word M_p . The key S , on the basis of which the substitution is effected, comprises n key bits $S[n-1]...S[0]$, wherein one of these key bits $S[n-1]...S[0]$ is supplied to each of the substitution elements. Substitution elements $15_n-1...15_0$ are designed, as determined by the respective substitution key bit $S[n-1]...S[0]$, to output in unchanged or inverted form the data bit $M_p[n-1]...M_p[0]$ supplied to the respective substitution element $15_n-1...15_0$.

A circuit-logic implementation of an embodiment of this substitution element is shown in Figure 9. The substitution element 15_k comprises a first and second AND gate $AND3$, $AND4$, and an OR gate $OR2$ connected following AND gates $AND3$, $AND4$, at the output of which OR gate the substituted data bit is provided. The substituted data bit is supplied to the substitution element through a first input $IN4$, and this data bit is supplied in inverted form by a first inverter $INV2$ to first AND gate $AND3$, and in unchanged form to second AND gate $AND4$. The respective substitution key applied at a second input $IN5$ of the substitution element is supplied to first gate $AND3$ in unchanged form, and to second AND gate $AND4$ in inverted form by a second inverter $INV3$. This arrangement ensures that given a substitution key bit "1" the data bit applied at first input $IN4$ is provided in inverted form, and given a substitution key bit "0" this data bit is provided in unchanged form at output $OUT2$.

In the embodiment of Figure 3, the encrypted data word M' is generated from unencrypted data word M by permutation and subsequent substitution of data word M_p resulting from the permutation. It is of course understood that it is also possible first to substitute data word M using substitution key M , and then to permute the resulting substituted data word using permutation key P in order to arrive at the encrypted data word M' .

The determining factor for the efficacy of an encryption system is the number of different possible keys. In the example described, key C to encrypt data word M is composed of permutation key P and substitution key S . Permutation key P comprises a number of subkeys corresponding to the number of data bits, the length of the subkeys being defined by $m=\log_2(n)$. With reference to Figure 10, the permutation key can be viewed as a vector with n subkeys $P[n-1]...P[0]$, or as an $n \times m$ matrix of individual subkey bits $P[n-1,m-1]...P[0,0]$. For data words of length $n=32$, the permutation key comprises 32 different subkeys $P[n-1]...P[0]$, thereby resulting in $32!$ different key combinations. Given that for substitution key S there are 2^n available possibilities, then for the number N possible keys C for data words to be encrypted of length $n=32$ the result is: $N = (32!) \cdot 2^{32}$.

Substitution key S for encryption and decryption can easily be generated as part of a binary random sequence.

A method of generating the permutation key is explained below for a data word of length $n=4$ bit based on Figures 11 through 13.

Figure 11 first shows a permutation unit 14 to generate permuted data word M_p from data word M with $n=4$ selection units $14_3, 14_2, 14_1, 14_0$ which are each of two-stage form ($m=\log_2 4=2$).

Figure 12 shows a second permutation unit corresponding to permutation unit 14 of Figure 11 which functions to undo the permutation effected by first permutation unit 14 as it decrypts the data word in the decryption unit (11 in Figure 3). This second permutation unit 14' is identical to first permutation unit 14 in structure and comprises four selection units $14'_3, 14'_2, 14'_1, 14'_0$. Each of these selection units $14'_3 \dots 14'_0$ functions to map one of data bits $M_p[3] \dots M_p[0]$ of permuted data word M_p back to one of data bits $M[3] \dots M[0]$ of original data word M . This selection of one of the data bits in individual selection units $14'_3 \dots 14'_0$ is effected in each case as determined by subkeys $P'[3] \dots P'[0]$ of a second permutation key P' , wherein in the example shown $P'=(P'[3], P'[2], P'[1], P'[0])$, the individual subkeys $P'[3] \dots P'[0]$ each comprising two subkey bits $P'[3,1] \dots P'[0,0]$.

The generation of subkeys $P[3] \dots P[0]$ of first permutation key P and of the associated subkeys $P'[3] \dots P'[0]$ of second permutation key P' is explained below based on Figure 13.

To generate the first and second permutation keys P, P' , the key generator (13 in Figure 2) comprises a first and second key memory 131, 131', as well as an assignment register¹ 132. Key memories 131, 131' are each designed to store n subkeys of key width $m=\log_2(n)$. Given $n=4$, four subkeys of length 2 are storable in each key memory. Assignment of the subkeys stored in first key memory 131 to selection units $14_3 \dots 14_0$, and thus to the individual data bits of permuted data word M_p , is effected through the address of key memory 131 which is addressable line-by-line and which in the example comprises $n=4$ lines. The memory address of a subkey in this first memory 131 corresponds here to the bit position of the data bit of the permuted data word to which the respective key is assigned. A subkey $P[k]$ at the memory address k of key memory 131 is thus assigned to the k^{th} data bit $M_p[k]$ of permuted data word M_p , where k represents one of the possible line addresses $0 \dots n-1$ of the memory.

¹ This is later called "assignment memory 132 and, in the list of reference numbers, "selection register." Translator.

Assignment of subkeys $P'[3]...P'[0]$ of second subkey P' to selection units $14'_3 ... 14'_0$ or to data bits $M[3]...M[0]$ of the original data word is effected analogously. That is, subkey $P'[k]$ stored at memory position k of second key memory 131' is assigned to selection unit $14'_k$ and determines which of the data bits of permuted data word M_p is to be mapped to data bit $M[k]$ at the k^{th} position of data word M .

Generation of subkeys $P[3]...P[0]$ of the first permutation key and of second subkeys $P'[3]...P'[0]$ is effected in a mutually matched fashion by a procedure which is explained below.

The subkeys of first permutation key P are generated consecutively as random binary sequences of length $m=2$ using the function generator 12 shown in Figure 2. As explained, the individual subkeys must differ from one another in order to obtain a one-to-one assignment of the data bits of data word to be permuted M to the data bits of permuted data word M_p . In the example described based on Figures 11 and 12, there are $n=4$ different subkeys which can be assigned randomly to the four selection units.

One memory position of assignment register 132 is assigned to each of the possible different subkeys, in this case, "11", "10", "01", "00", wherein a predetermined value is entered in the assignment register at the respective position if the assigned subkey has already been generated at a memory position of memory 131, and thus for one of selection units $14_3...14_0$, so as to avoid again generating the same key at a different memory address, and thus for another selection unit $14_3...14_0$.

In the example, the assignment of a certain one of the possible subkeys to a memory address of assignment register 132 is effected by directly mapping the value represented by the subkey to the address of the memory position of mapping memory 132. For example, the memory position $10_2=2$ of assignment memory 132 is thus assigned to a subkey "10". If $P[k]=w_{n-1} ... w_0$ applies for a subkey, then for the address assigned to this subkey:

$$W = \sum_{i=0}^{i=n-1} w_i 2^i$$

In order to generate the permutation key, the respective subkeys are randomly generated consecutively for the individual memory addresses of first permutation key memory 131,

wherein after generation of a given subkey a determination is made based on examination of the assignment register whether such a subkey has already been generated. If such a subkey has already been generated, the subkey is rejected and a new subkey is randomly generated. This procedure is repeated until subkeys have been generated for all the memory positions, and thus for all the selection units of permutation unit 14.

When one of the possible subkeys is generated for the first time, a certain value, for example a "1," is entered at the memory address, assigned to this key, of assignment memory 132. If this subkey is randomly generated once again for another memory position of memory 131, this is detected in assignment memory 132 based on the value entered, and the subkey is rejected for this different memory position.

As explained above, the binary value of a subkey $P[3]...P[0]$ which is assigned to a selection unit $14_3...14_0$ or to a data bit $Mp[3]...Mp[0]$ of permuted data word Mp corresponds to the data position of the data bit $M[3]...M[0]$ of the input word M selected by the respective selection unit. Accordingly, subkeys $P'[n-1]...P'[0]$ of second permutation key P' each indicate which of the data bits of permuted data word Mp is to be mapped to data bit $M[3]...M[0]$ to which the respective subkey is assigned.

If the general condition applies that a subkey $P[k]$ assigned to the k^{th} data bit $Mp[k]$ of permuted data word Mp maps the i^{th} data bit $M[i]$ of the permuted data word to this data bit of permuted data word Mp , then, conversely, the subkey $P'[i]$ assigned to the i^{th} data bit must map the k^{th} data bit of permuted data word Mp to this data bit.

Second key memory 131' is organized analogously to first key memory 131, that is, the addresses at which the individual subkeys $P'[n-1]...P'[0]$ are stored correspond to the bit positions of the data bits $M[n-1]...M[0]$ to which the individual subkeys are assigned.

In order to generate a matching subkey of second permutation key P' for a randomly generated subkey $P[k]$ of first permutation key P , which subkey is assigned to the k^{th} data bit of permuted data word Mp , the address value k of first subkey $P[k]$ is entered at the address in second key memory 131', the value of which corresponds to the binary value i represented by the first key. In other words, for $P[k]=i$, $P'[i]=k$.

Generation of the first and second permutation keys can be described by the following algorithm:

Line 1: FOR $k = (n-1)$ DOWNT0 0

Line 2: Fetch random number from generator and compute i

Line 3: Check if $\text{MapReg}(i) = 1$, if true, go to Line 2

Line 4: Set $\text{MapReg}(i) = 1$

Line 5: Set $o_store(k) = i$

Line 6: Set $i_store(i) = k$

Line 4: NEXT k .

$\text{MapReg}(i)$ here represents the value at address k of the assignment register. The expression $o_store(k)$ represents the value at address k of the first memory, while $i_store(i)$ represents the value at address i of second memory [31].

As explained above, the permutation effected during encryption and analogously during decryption is augmented by a substitution as determined by a substitution key. This substitution can be effected either before the permutation or after the permutation, the procedure being effected in the reverse order during the decryption. If during encryption the substitution is effected after the permutation, then during decryption the re-substitution is effected before the permutation. During the above-described substitution in which, as determined by the substitution key bits, the respective assigned data bit is passed on either inverted or unchanged, the same substitution key used during decryption is used during encryption.

List of reference notations

AND1-AND4	AND-gate
C,C'	key
IN1-IN5	inputs
INV1, INV2	inverter
M	data word
M[n-1]... M[0]	data bits
M'[n-1]...M'[0]	data bits of an encrypted data word
Mp[n-1]...Mp[0]	data bits of a permuted data word
OR1, OR2	OR-gate
OUT1, OUT2	outputs
P	permutation key
P[n-1]...P[0]	subkey of a permutation key
S	substitution key tes ²
10	encryption and decryption unit
11	encryption unit
11'	decryption unit
13	key generator
14	permutation unit
14_n-1...14_0	selection unit
15	substitution unit
15_n-1...15_0	substitution units
20	random access memory, RAM
20	random number generator
21	input of the RAM
22	output of the RAM
30	data processing unit
110	input of the encryption unit
110'	input of the decryption unit

² Translators note: this does not appear to belong.

111	output of the encryption unit
111'	output of the decryption unit
112	key input of the encryption unit
112'	key input of the decryption unit
131	first permutation key memory
131'	second permutation key memory
132	selection register
141_n-1... 141_0	selection stages
142	selection switch